Application project

Power 4++

Gilles Venturini, venturini@univ-tours.fr, office 204/220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

Objectives of this project (18H, 9 sessions)

- Produce a Power 4/Connect 4/Puissance 4 game
- With a GOFAI method called the min-max algorithm
- With special assets (bombs, …, even a Pac-Man eater!)

- Use all notions learned in Algorithms, Software engineering and GUI courses

- Produce: Python code (testable game) + report with 3 parts
  - Algorithms: main data structures and algorithms
  - Software engineering: technical documentation about code organization
  - GUI (user needs, mockups, prototype, user evaluation)

https://lululataupe.com/tout-age/686-puissance-4

Functionnal requirements

- Two players (human and computer), variable board size, variable nb of aligned coins to win, 2 colors for coins (= more or less the standard game)
- Select coins color + game level + who is playing first (computer or human, random choice)
- Special tricks/assets (at least one): bomb (clear column partially or totally), …, Flip (Upside down flipping of the board), your ideas (but keep it simple = deterministic behavior + no more than 1 parameter = one column or row etc)
- Several assets possible but each asset is used only once in a game (?)
- GUI with Undo/redo
- Use of the min-max algorithm.
- Bonus: add player advices + explain the AI strategy to the user (in the GUI).

Non functionnal requirements

- Use concepts learned in Software Engineering, Algorithms and GUI courses
- Software Engineering course:
  - separability, abstraction, weak coupling, uniformity, encapsulation,
  - naming convention, code factorization, precise typing,
- Algorithms course:
  - clear definition of data structures and main algorithms, specifications (comment the code with inputs, preconditions, outputs, postconditions)
  - Avoid using “breaks”
- GUI course:
  - User needs (problem modeling, user profile(s), task models, constraints)
  - GUI specification: pencil and paper, mockup, prototype
  - GUI implementation: Python code with TKinter
  - GUI test and evaluation: user evaluation

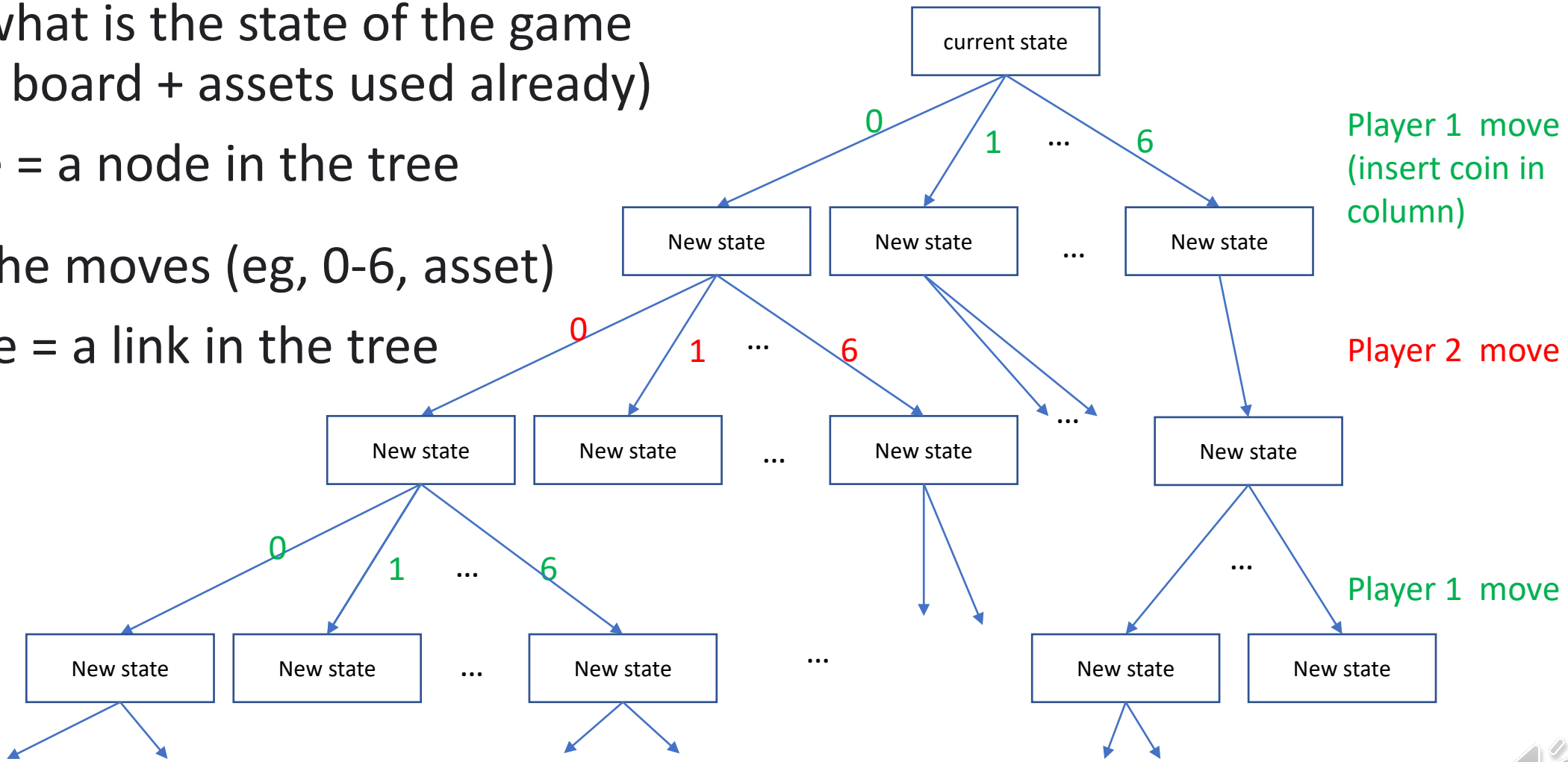
Game tree

- Define what is the state of the game (current board + assets used already)

=> A state = a node in the tree

- Define the moves (eg, 0-6, asset)

=> A move = a link in the tree

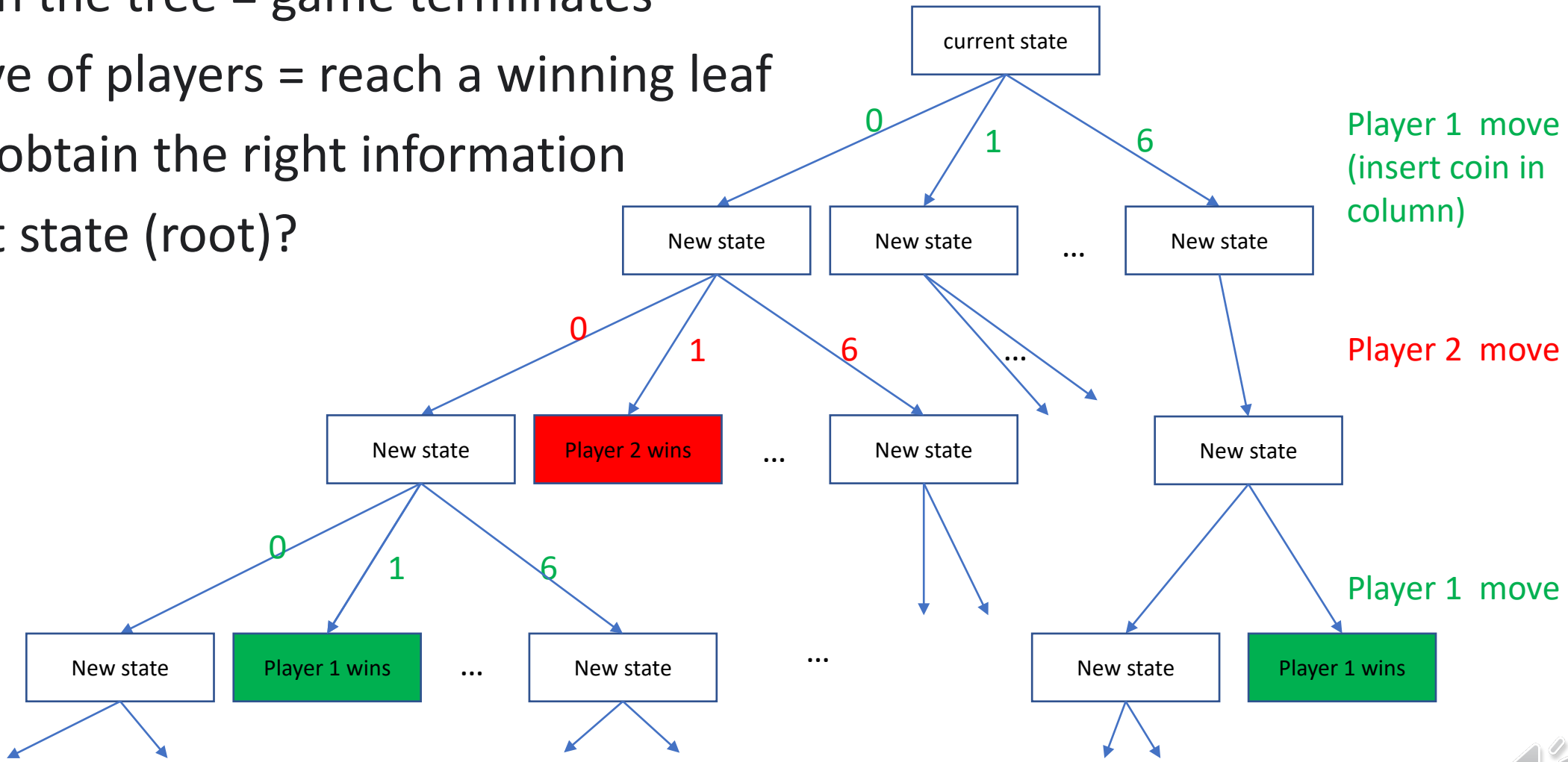


https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

Game tree

- Leaves in the tree = game terminates
- Objective of players = reach a winning leaf
- How to obtain the right information

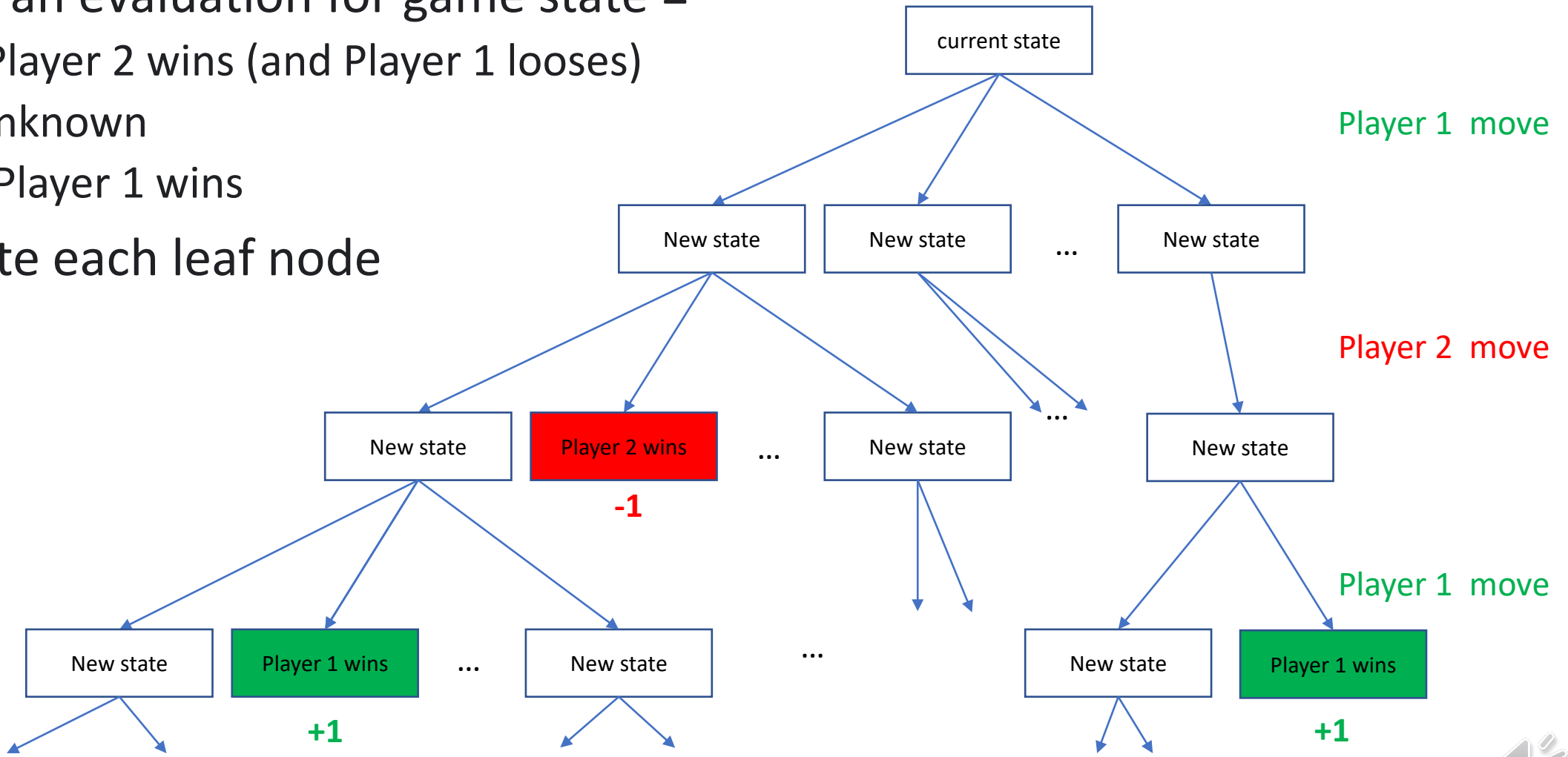
at current state (root)?



https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

From game tree to Min-Max

- Define an evaluation for game state =
  - -1, Player 2 wins (and Player 1 looses)
  - 0, unknown
  - +1, Player 1 wins
- Evaluate each leaf node

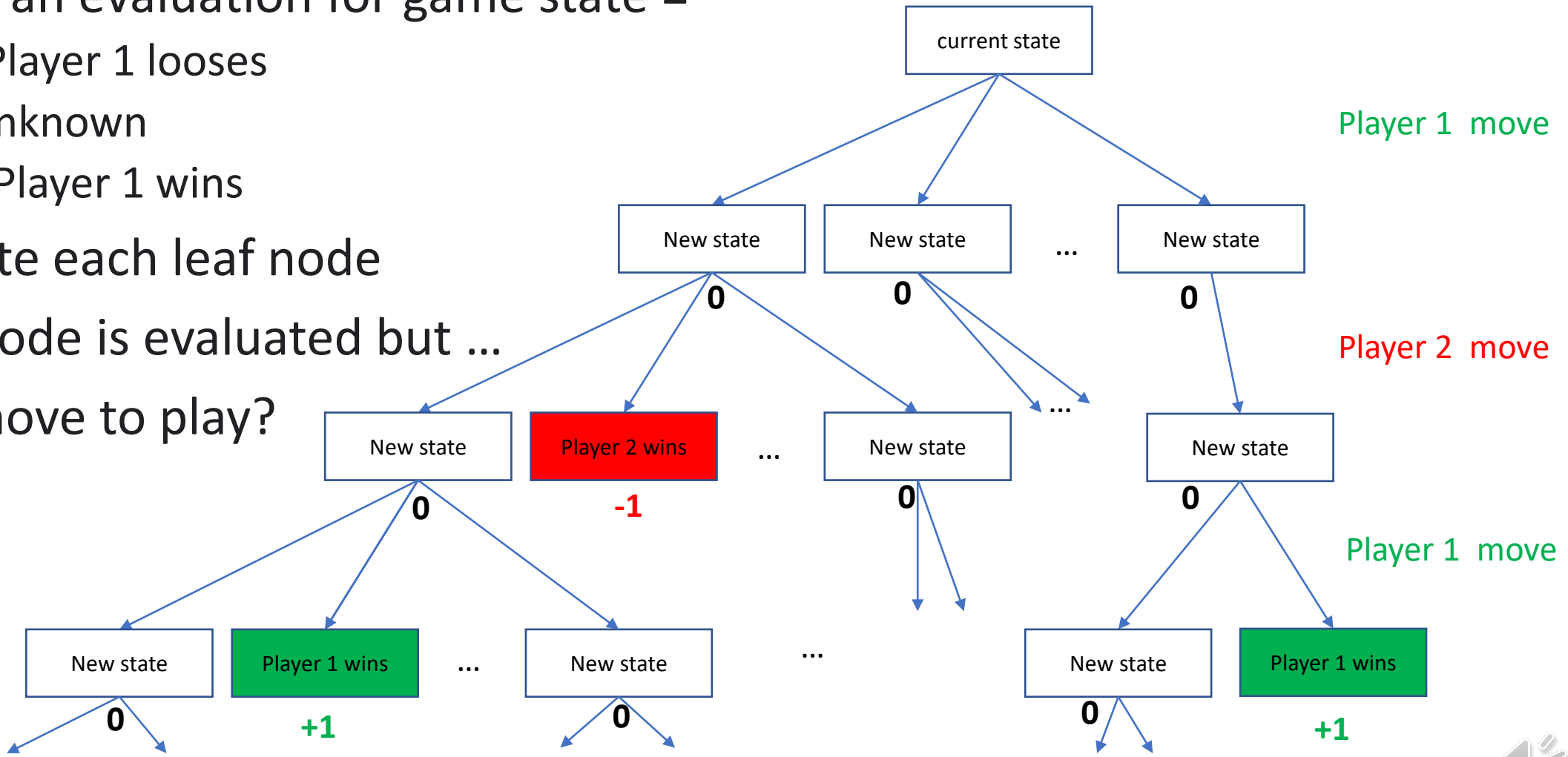


https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

From game tree to Min-Max

- Define an evaluation for game state =
  - -1, Player 1 looses
  - 0, unknown
  - +1, Player 1 wins
- Evaluate each leaf node
- Each node is evaluated but …

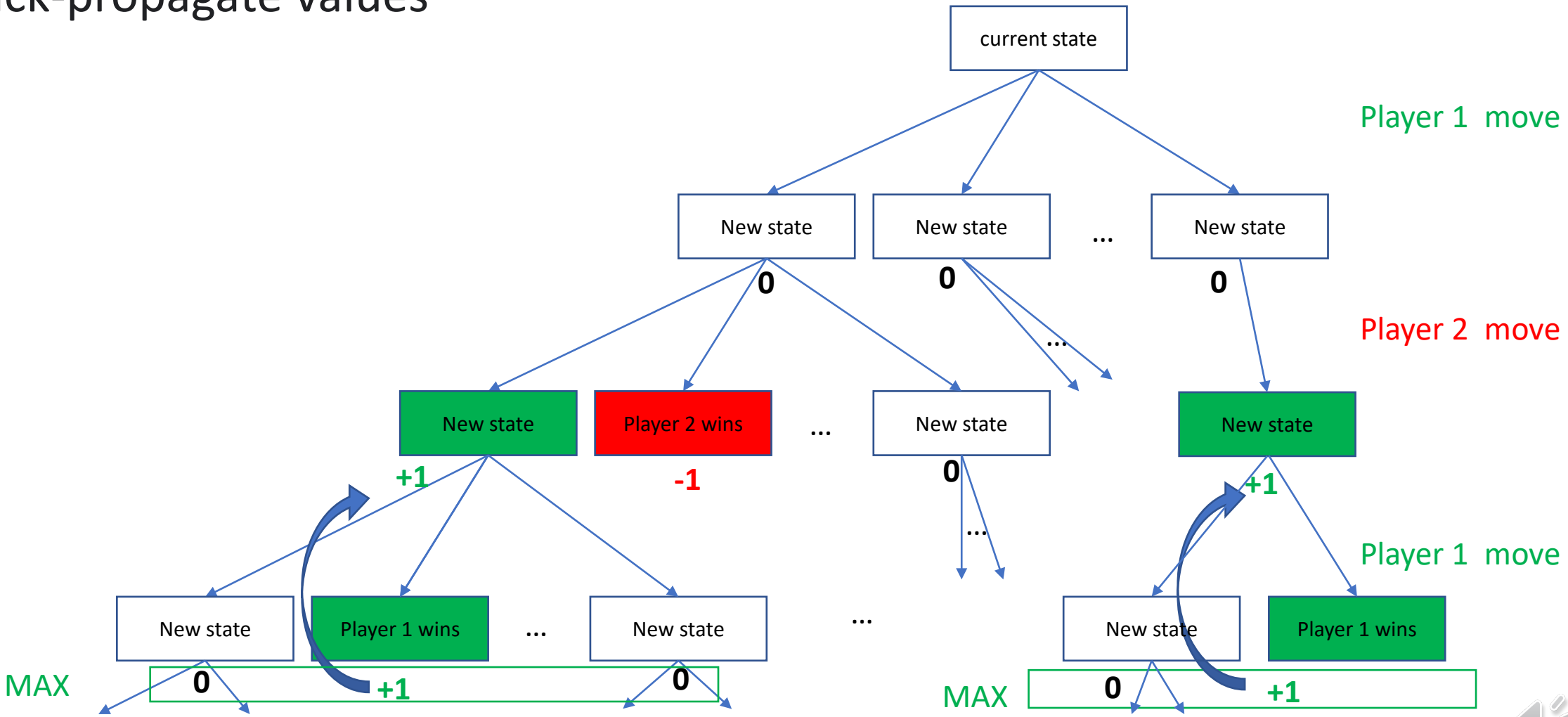
which move to play?



https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

Min-Max algorithm

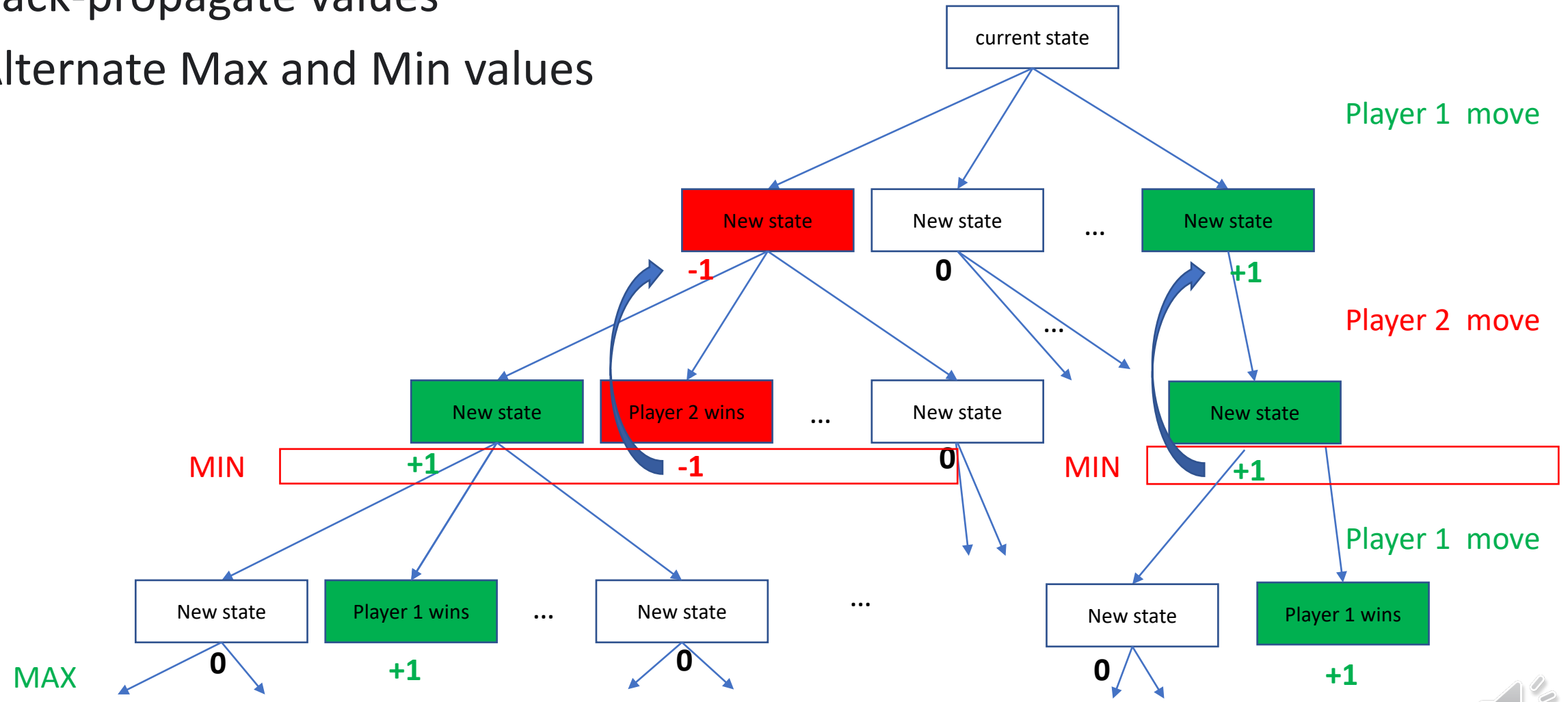
- Back-propagate values



https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

Basic Min-Max

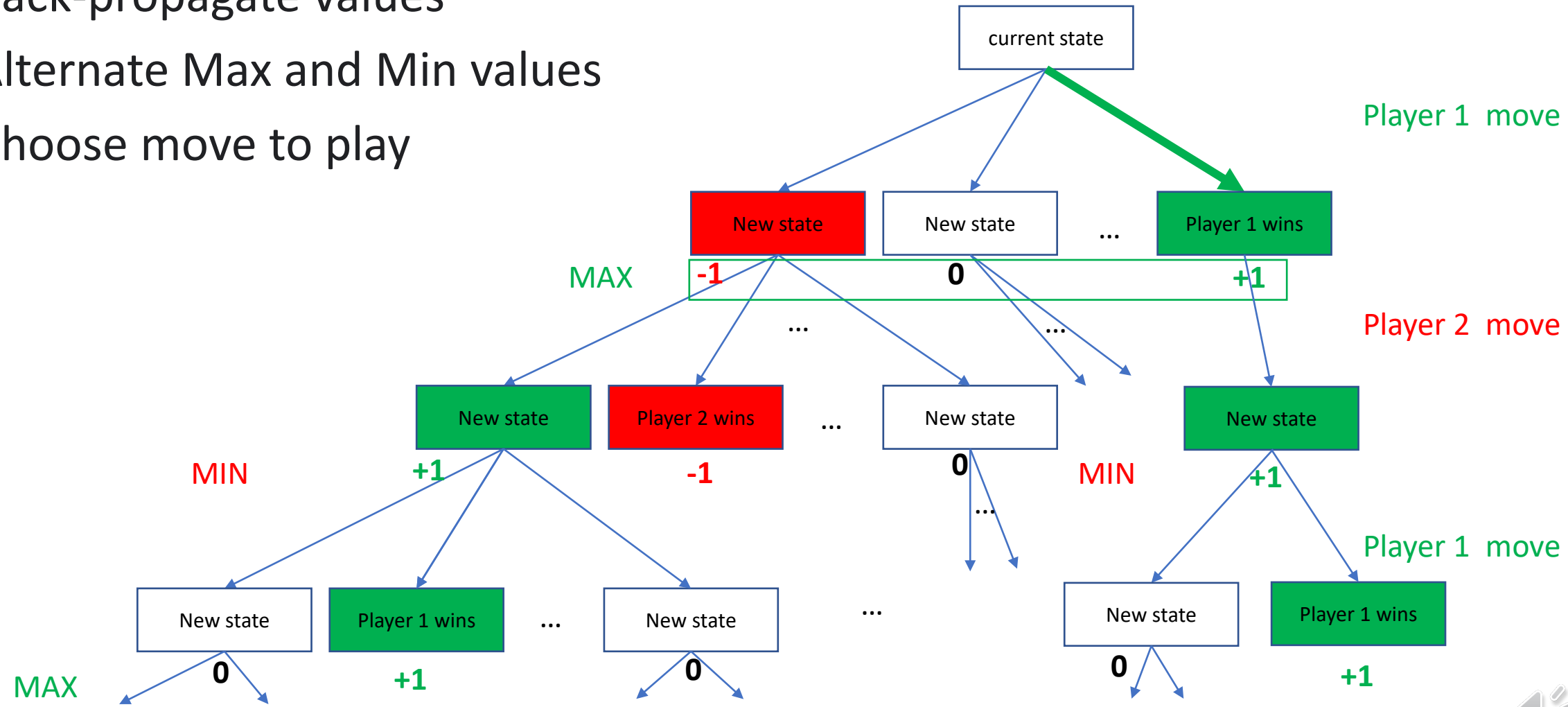
- Back-propagate values
- Alternate Max and Min values



https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

Basic Min-Max

- Back-propagate values
- Alternate Max and Min values
- Choose move to play



https://en.wikipedia.org/wiki/Game_tree https://fr.wikipedia.org/wiki/Algorithme_minimax

Evaluation of states

- Infinite or defined values for leaves:
  - +1 or +∞ if you win
  - -1, 0 or -∞ if you loose

- Not just « binary values » but continuous values if possible
- Between victory and defeat: find a smart game state evaluation (predictive enough about victory and defeat)
- What is an interesting game state for you? (in the case of P4++)

- Min-Max => recursive algorithm with a maximum depth for tree exploration
- Maximum depth = game level (deep game tree = high computational cost)

Outcome of your project

- See the 3 courses (Software Engineering, Algorithms, GUIs)

- Commented Python code
- Report with 3 parts
  - main data structures and algorithms
  - technical documentation about code organization
  - GUI (from User needs, mockups, prototype, user evaluation)

- To be uploaded on Celene

Other hints for project development

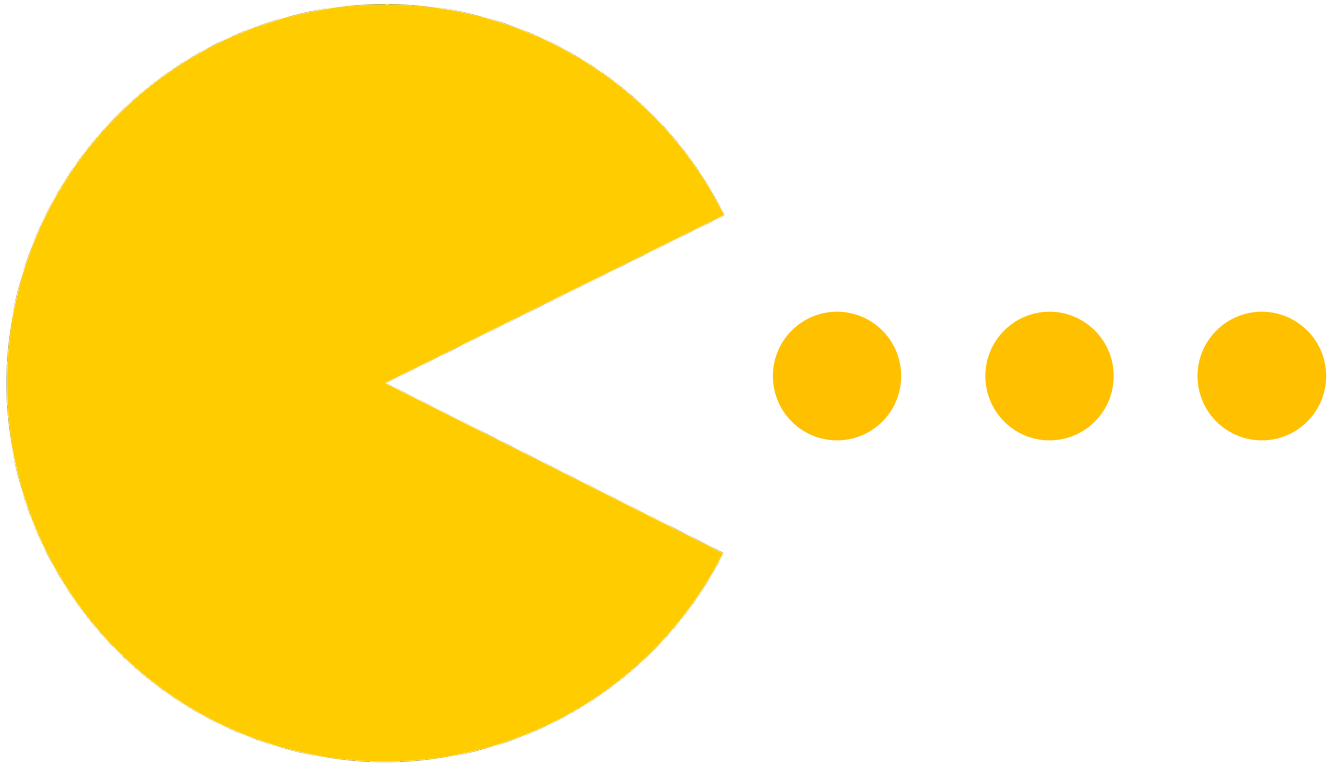
- Start with needs analysis, algorithms, program structure, GUI specifications and then start implementing

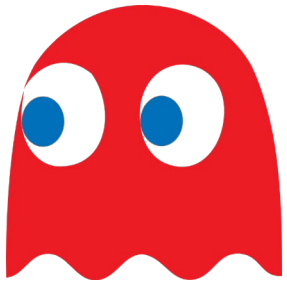
- About the programming process:
  - Begin with "text/console" interface to get core data structures and algorithms right, then study the GUI
  - Begin with random strategy for computer player and then implement Min Max.
  - Begin with standard moves (one coin in a given column) then add assets

- Clear programming with Python
  - One line of code = one line of comment (yes you can!)
  - Avoid using classes (OO course coming soon)
  - Use TypeHint

https://docs.python.org/3/library/typing.html

Good luck!





Domaine public, https://commons.wikimedia.org/w/index.php?curid=667189